Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Петрозаводский государственный университет» Физико-технический институт Кафедра информационно-измерительных систем и физической электроники

ПРАКТИЧЕСКАЯ РАБОТА №1

ОТЧЁТ

по предмету «Обработка цифровых сигналов»

Пространственные методы улучшения и восстановления

изображений.

Автор работы: студент группы 21512 Н. Ю. Новохатько \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Научный руководитель: доцент А.В. Бульба

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Петрозаводск 2020

**Результаты выполненной работы:**

*Назначение программы:*

Демонстрация обработки изображения от униполярного и биполярного шумов фильтрами среднегармонического и медианным.

*Код программы:*

*Листинг main.cpp*

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(*argc*, argv);

MainWindow w;

w.show();

return a.exec();

}

*Листинг mainwindow.h*

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <shumform.h>

namespace **Ui** {

class **MainWindow**;

}

class **MainWindow** : public QMainWindow

{

Q\_OBJECT

public:

explicit **MainWindow**(QWidget \*parent = nullptr);

~***MainWindow***();

private:

Ui::MainWindow \*ui;

QImage Ishodnoe;

QImage Zashumlennoye\_one;

QImage Zashumlennoye\_two;

QImage RShum;

QImage GShum;

QImage BShum;

QImage Obrabotannoye;

ShumForm \*SF1;

public slots:

void **UniImage**();

void **BiImage**();

void **MedUniImage**();

void **MedBiImage**();

//void NonShumImage();

//void BackFromSignalShum(QImage Ishodnoe2); //, QImage Zashumlennoye\_one2, QImage Zashumlennoye\_two2);

//void testslot();

};

#endif // MAINWINDOW\_H

*Листинг shumform.h*

#ifndef SHUMFORM\_H

#define SHUMFORM\_H

#include <QWidget>

#include <QImage>

#include <QVector>

namespace **Ui** {

class **ShumForm**;

}

class **ShumForm** : public QWidget

{

Q\_OBJECT

public:

explicit **ShumForm**(QWidget \*parent = nullptr,

QString imageWay = "",

double imageSigma = 0,

int imageMask = 0,

bool noise\_flag = false,

bool filter\_flag = false);

~***ShumForm***();

private:

Ui::ShumForm \*ui;

QString imageWay;

double imageSigma;

int imageMask;

QVector <QImage> iImage;

public slots:

//Сохранее отмеченных изображений

void **saveImages**();

};

#endif // SHUMFORM\_H

*Листинг mainwindow.cpp*

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QDebug>

#include <QImage>

#include <QPainter>

#include <shumform.h>

MainWindow::**MainWindow**(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

connect(ui->pushButton\_2, SIGNAL(clicked()), this, SLOT(UniImage()) );

connect(ui->pushButton\_3, SIGNAL(clicked()), this, SLOT(BiImage()) );

connect(ui->pushButton\_4, SIGNAL(clicked()), this, SLOT(MedUniImage()) );

connect(ui->pushButton\_5, SIGNAL(clicked()), this, SLOT(MedBiImage()) );

//connect(SF1, SIGNAL(signalBack(QImage) ), this, SLOT(BackFromSignalShum(QImage) ));

}

MainWindow::~***MainWindow***()

{

delete ui;

}

void MainWindow::**UniImage**()

{

qDebug() << "ShumImage";

QString imageWay = ui->lineEdit->text(); qDebug() << "imageWay = " << imageWay;

QString imageSigma1 = ui->lineEdit\_2->text(); //qDebug() << "imageSigma = " << imageSigma1;

QString imageMask1 = ui->lineEdit\_3->text(); //qDebug() << "imageMask = " << imageMask1;

double imageSigma = imageSigma1.toDouble(); qDebug() << "imageSigma = " << imageSigma;

int imageMask = imageMask1.toInt(); qDebug() << "imageMask = " << imageMask;

bool flag = false; //???

bool flag1 = false; //???

//Присвоение адреса объекта дочернего окна ссылке

SF1 = new ShumForm(nullptr, imageWay, imageSigma, imageMask, flag, flag1);

SF1->show();

}

void MainWindow::**BiImage**()

{

qDebug() << "BiImage";

QString imageWay = ui->lineEdit->text(); qDebug() << "imageWay = " << imageWay;

QString imageSigma1 = ui->lineEdit\_2->text(); //qDebug() << "imageSigma = " << imageSigma1;

QString imageMask1 = ui->lineEdit\_3->text(); //qDebug() << "imageMask = " << imageMask1;

double imageSigma = imageSigma1.toDouble(); qDebug() << "imageSigma = " << imageSigma;

int imageMask = imageMask1.toInt(); qDebug() << "imageMask = " << imageMask;

bool flag = true; //???

bool flag1 = false; //Включение фильтра среднего контргармонического

//Присвоение адреса объекта дочернего окна ссылке

SF1 = new ShumForm(nullptr, imageWay, imageSigma, imageMask, flag, flag1);

SF1->show();

}

void MainWindow::**MedUniImage**()

{

qDebug() << "MedUniImage";

QString imageWay = ui->lineEdit->text(); qDebug() << "imageWay = " << imageWay;

QString imageMask1 = ui->lineEdit\_4->text(); //qDebug() << "Med imageMask = " << imageMask1;

double imageSigma = 0; //imageSigma1.toDouble(); qDebug() << "imageSigma = " << imageSigma;

int imageMask = imageMask1.toInt(); qDebug() << "imageMask = " << imageMask;

bool flag = false; //???

bool flag1 = true; //Включение медианного фильтра

//Присвоение адреса объекта дочернего окна ссылке

SF1 = new ShumForm(nullptr, imageWay, imageSigma, imageMask, flag, flag1);

SF1->show();

}

void MainWindow::**MedBiImage**()

{

qDebug() << "MedUniImage";

QString imageWay = ui->lineEdit->text(); qDebug() << "imageWay = " << imageWay;

QString imageMask1 = ui->lineEdit\_4->text(); //qDebug() << "Med imageMask = " << imageMask1;

double imageSigma = 0; //imageSigma1.toDouble(); qDebug() << "imageSigma = " << imageSigma;

int imageMask = imageMask1.toInt(); qDebug() << "imageMask = " << imageMask;

bool flag = true; //???

bool flag1 = true; //Включение медианного фильтра

//Присвоение адреса объекта дочернего окна ссылке

SF1 = new ShumForm(nullptr, imageWay, imageSigma, imageMask, flag, flag1);

SF1->show();

}

/\*

void MainWindow::NonShumImage()

{

qDebug() << "NonShumImage";

}

void MainWindow::BackFromSignalShum(QImage Ishodnoe2)//, QImage Zashumlennoye\_one2, QImage Zashumlennoye\_two2)

{

Ishodnoe = Ishodnoe2; qDebug() << "Ishodnoe - сохранено!";

Zashumlennoye\_one = Zashumlennoye\_one2; qDebug() << "Zashumlennoye\_one - сохранено!";

Zashumlennoye\_two = Zashumlennoye\_two2; qDebug() << "Zashumlennoye\_two - сохранено!";

}

void MainWindow::testslot()

{

}

\*/

*Листинг shumform.cpp*

#include "shumform.h"

#include "ui\_shumform.h"

#include <QPainter>

#include <QLabel>

#include <QPixmap>

#include <QDebug>

#include <QColor>

#include <QRandomGenerator>

#include <QImage>

#include <QtMath>

#include <QCheckBox>

ShumForm::**ShumForm**(QWidget \*parent, QString imageWay, double imageSigma, int imageMask, bool noise\_flag, bool filter\_flag) :

QWidget(parent),

ui(new Ui::ShumForm)

{

ui->setupUi(this);

//QPixmap pix("D:/Obrazets/1.bmp");

//QImage image("D:/Obrazets/1.bmp");

//ui->label->setPixmap(pix);

//QPixmap pix = QPixmap->fromImage(pix);

//QString fileName = "D:/Obrazets/1test.bmp";

QString fileName = imageWay;

if (!fileName.isEmpty())

{

QImage image(fileName);

if (image.isNull())

{

qDebug() << "Error load image";

}

ui->label->setPixmap(QPixmap::fromImage(image));

iImage.append(image);

int iwidth = image.width(); qDebug() << "[ширина] width = " << iwidth;

int iheight = image.height(); qDebug() << "[высота] iheight = " << iheight;

int w,h;

int s = int(iwidth\*iheight\*0.1); qDebug() << "[процент зашумлённых точек от общего их количества] s = " << s;

//Создание объекта "Цвет пикселя".

QColor white = QColor(255,255,255,0);

QColor black = QColor(0,0,0,0);

//Создание объекта точка

QPoint p, q;

bool flag = true;

QImage clone\_oneside = image;

if (noise\_flag == false)

{

//однополярный шум

for (int i = 0; i < s; i++)

{

w = int (QRandomGenerator::global()->generateDouble() \* iwidth); //qDebug() << "[Произвольное положение по w] w = " << w;

h = int (QRandomGenerator::global()->generateDouble() \* iheight); //qDebug() << "[Произвольное положение по h] h = " << h;

p = QPoint(w, h);

clone\_oneside.setPixelColor(p, white);

}

}

else

{

//двуполярный шум

for (int i = 0; i < s; i++)

{

w = int (QRandomGenerator::global()->generateDouble() \* iwidth); //qDebug() << "[Произвольное положение по w] w = " << w;

h = int (QRandomGenerator::global()->generateDouble() \* iheight); //qDebug() << "[Произвольное положение по h] h = " << h;

p = QPoint(w, h);

if (flag == true)

{

clone\_oneside.setPixelColor(p, black);

flag = false;

}

else

{

clone\_oneside.setPixelColor(p, white);

flag = true;

}

}

qDebug() << "двуполярный шум" ;

}

ui->label\_2->setPixmap(QPixmap::fromImage(clone\_oneside));

iImage.append(clone\_oneside);

QColor iRgb, iredd, igreenn, ibluee;

int ired, igreen, iblue;

QImage clone\_oneside\_red = image;

QImage clone\_oneside\_green = image;

QImage clone\_oneside\_blue = image;

for (int i = 0; i < iheight; i++)

{

for(int j = 0; j < iwidth; j++)

{

p = QPoint(j, i);

iRgb = clone\_oneside.pixelColor(p);

ired = iRgb.red(); //qDebug() << "[] ired = " << ired;

iredd = QColor(ired, 0, 0, 0);

clone\_oneside\_red.setPixelColor(p, iredd);

igreen = iRgb.green();

igreenn = QColor(0, igreen, 0, 0);

clone\_oneside\_green.setPixelColor(p, igreenn);

iblue = iRgb.blue();

ibluee = QColor(0, 0, iblue, 0);

clone\_oneside\_blue.setPixelColor(p, ibluee);

}

}

ui->label\_4->setPixmap(QPixmap::fromImage(clone\_oneside\_red));

ui->label\_5->setPixmap(QPixmap::fromImage(clone\_oneside\_green));

ui->label\_6->setPixmap(QPixmap::fromImage(clone\_oneside\_blue));

iImage.append(clone\_oneside\_red);

iImage.append(clone\_oneside\_green);

iImage.append(clone\_oneside\_blue);

//Ширина маски

int mask\_width = imageMask;

//Высота маски

int mask\_height = imageMask;

//радиус маски (если квадратная)

int mask\_radius\_width, mask\_radius\_height;

//Начальные положения для прохода по маске

int i0, j0;

mask\_radius\_width = (mask\_width - 1) / 2; qDebug() << "[] mask\_radius\_width = " << mask\_radius\_width;

mask\_radius\_height = (mask\_height - 1) / 2; qDebug() << "[] mask\_radius\_height = " << mask\_radius\_height;

//Коэффициент сигма

double sigma;

//Значение пикселя в маске, определяемое по ширине и высоте маски

double pix\_mask\_red, pix\_mask\_green, pix\_mask\_blue;

double pix\_mask\_red\_0, pix\_mask\_green\_0, pix\_mask\_blue\_0;

double pix\_mask\_red\_1, pix\_mask\_green\_1, pix\_mask\_blue\_1;

pix\_mask\_red = 0;

pix\_mask\_green = 0;

pix\_mask\_blue = 0;

pix\_mask\_red\_0 = 0;

pix\_mask\_green\_0 = 0;

pix\_mask\_blue\_0 = 0;

pix\_mask\_red\_1 = 0;

pix\_mask\_green\_1 = 0;

pix\_mask\_blue\_1 = 0;

if (filter\_flag == false)

{

sigma = imageSigma;

//Перебор с формированием маски, создание обработанных одноканальных изображений

for (int i = mask\_radius\_height; i < (iheight - mask\_radius\_height); i++) //mask\_radius\_height; i < (iheight - mask\_radius\_height + 1); i++)

{

for(int j = mask\_radius\_width; j < (iwidth - mask\_radius\_width); j++) //mask\_radius\_width; j < (iwidth - mask\_radius\_width + 1); j++)

{

p = QPoint(j, i);

i0 = i - mask\_radius\_height; //qDebug() << "[Координата начала маски] i0 = " << i0;

j0 = j - mask\_radius\_width; //qDebug() << "[Координата начала маски] j0 = " << j0;

for (int k = i0; k < (i0 + mask\_height); k++) //i0; k < mask\_height; k++)

{

for(int m = j0; m < (j0 + mask\_width); m++) //j0; m < mask\_width; m++)

{

q = QPoint(m, k); //qDebug() << "[Точка] q = " << q;

iRgb = clone\_oneside\_red.pixelColor(q); //qDebug() << "[!!!] iRgb.red() = " << double(iRgb.red()); //qDebug() << "[!!!] iRgb = " << iRgb;

if (iRgb.red() > 0)

{

pix\_mask\_red\_0 += qPow(double(iRgb.red()), sigma); //qDebug() << "[QQQ] pix\_mask\_red\_0 = " << pix\_mask\_red\_0;

pix\_mask\_red\_1 += qPow(double(iRgb.red()), sigma + 1); //qDebug() << "[QQQ] pix\_mask\_red\_1 = " << pix\_mask\_red\_1;

}

iRgb = clone\_oneside\_green.pixelColor(q); //qDebug() << "[!!!] iRgb.green() = " << iRgb.green();// qDebug() << "[!!!] iRgb = " << iRgb;

if (iRgb.green() > 0)

{

pix\_mask\_green\_0 += qPow(double(iRgb.green()), sigma);

pix\_mask\_green\_1 += qPow(double(iRgb.green()), sigma + 1);

}

iRgb = clone\_oneside\_blue.pixelColor(q);

if (iRgb.blue() > 0)

{

pix\_mask\_blue\_0 += qPow(double(iRgb.blue()), sigma);

pix\_mask\_blue\_1 += qPow(double(iRgb.blue()), sigma + 1);

}

}

}

pix\_mask\_red = pix\_mask\_red\_1 / pix\_mask\_red\_0; //qDebug() << "[Результирующее значение в маске] pix\_mask\_red\_1 / pix\_mask\_red\_0 = " << pix\_mask\_red\_1

//<< pix\_mask\_red\_0;

pix\_mask\_green = pix\_mask\_green\_1 / pix\_mask\_green\_0; //qDebug() << "[Результирующее значение в маске] pix\_mask\_green = " << pix\_mask\_green;

pix\_mask\_blue = pix\_mask\_blue\_1 / pix\_mask\_blue\_0; //qDebug() << "[Результирующее значение в маске] pix\_mask\_blue = " << pix\_mask\_blue;

//Добавление точки в одноканальные изображения

//qDebug() << "[1]";

if ((0 <= pix\_mask\_red) && (pix\_mask\_red <= 256))

{

iredd = QColor(int(pix\_mask\_red), 0, 0, 0); //qDebug() << "[] iredd = " << iredd;

clone\_oneside\_red.setPixelColor(p, iredd);

}

//qDebug() << "[2]";

if ((0 <= pix\_mask\_green) && (pix\_mask\_green <= 256))

{

igreenn = QColor(0, int(pix\_mask\_green), 0, 0); //qDebug() << "[] igreenn = " << igreenn;

clone\_oneside\_green.setPixelColor(p, igreenn);

}

//qDebug() << "[3]";

if ((0 <= pix\_mask\_blue) && (pix\_mask\_blue <= 256))

{

ibluee = QColor(0, 0, int(pix\_mask\_blue), 0); //qDebug() << "[] ibluee = " << ibluee;

clone\_oneside\_blue.setPixelColor(p, ibluee);

}

pix\_mask\_red = 0;

pix\_mask\_green = 0;

pix\_mask\_blue = 0;

pix\_mask\_red\_0 = 0;

pix\_mask\_green\_0 = 0;

pix\_mask\_blue\_0 = 0;

pix\_mask\_red\_1 = 0;

pix\_mask\_green\_1 = 0;

pix\_mask\_blue\_1 = 0;

}

}

}

else

{

qDebug() << "Медианный фильтр!!!";

sigma = mask\_width \* mask\_height;

int buf;

QVector <int> pix\_mask\_red\_vect, pix\_mask\_green\_vect, pix\_mask\_blue\_vect;

//Перебор с формированием маски, создание обработанных одноканальных изображений

for (int i = mask\_radius\_height; i < (iheight - mask\_radius\_height - 1); i++) //mask\_radius\_height; i < (iheight - mask\_radius\_height + 1); i++)

{

for(int j = mask\_radius\_width; j < (iwidth - mask\_radius\_width - 1); j++) //mask\_radius\_width; j < (iwidth - mask\_radius\_width + 1); j++)

{

p = QPoint(j, i);

i0 = i - mask\_radius\_height; //qDebug() << "[Координата начала маски] i0 = " << i0;

j0 = j - mask\_radius\_width; //qDebug() << "[Координата начала маски] j0 = " << j0;

for (int k = i0; k < (i0 + mask\_height); k++) //i0; k < mask\_height; k++)

{

for(int m = j0; m < (j0 + mask\_width); m++) //j0; m < mask\_width; m++)

{

q = QPoint(m, k); //qDebug() << "[Точка] q = " << q;

iRgb = clone\_oneside\_red.pixelColor(q);

pix\_mask\_red\_vect.append(iRgb.red());

iRgb = clone\_oneside\_green.pixelColor(q);

pix\_mask\_green\_vect.append(iRgb.green());

iRgb = clone\_oneside\_blue.pixelColor(q);

pix\_mask\_blue\_vect.append(iRgb.blue());

}

}

//qDebug() << "[Результирующий массив] pix\_mask\_red\_vect = " << pix\_mask\_red\_vect;

//qDebug() << "[Результирующий массив] pix\_mask\_green\_vect = " << pix\_mask\_green\_vect;

//qDebug() << "[Результирующий массив] pix\_mask\_blue\_vect = " << pix\_mask\_blue\_vect;

//Сортировка массивов пузырьком. Красная составляющая.

for (int f = 0; f < sigma - 2; f++)

{

for (int t = f + 1; t < sigma - 1; t++)

{

if (pix\_mask\_red\_vect.at(f) > pix\_mask\_red\_vect.at(t))

{

buf = pix\_mask\_red\_vect.at(f);

pix\_mask\_red\_vect.replace(f, pix\_mask\_red\_vect.at(t));

pix\_mask\_red\_vect.replace(t, buf);

}

if (pix\_mask\_green\_vect.at(f) > pix\_mask\_green\_vect.at(t))

{

buf = pix\_mask\_green\_vect.at(f);

pix\_mask\_green\_vect.replace(f, pix\_mask\_green\_vect.at(t));

pix\_mask\_green\_vect.replace(t, buf);

}

if (pix\_mask\_blue\_vect.at(f) > pix\_mask\_blue\_vect.at(t))

{

buf = pix\_mask\_blue\_vect.at(f);

pix\_mask\_blue\_vect.replace(f, pix\_mask\_blue\_vect.at(t));

pix\_mask\_blue\_vect.replace(t, buf);

}

}

}

//qDebug() << "[Обработанный результирующий массив] pix\_mask\_red\_vect = " << pix\_mask\_red\_vect;

//Добавление точки в одноканальные изображения

//qDebug() << "[1]";

iredd = QColor(pix\_mask\_red\_vect.at( (int(sigma) + 1) / 2), 0, 0, 0); //qDebug() << "[] iredd = " << iredd;

clone\_oneside\_red.setPixelColor(p, iredd);

//qDebug() << "[2]";

igreenn = QColor(0, pix\_mask\_green\_vect.at( (int(sigma) + 1) / 2), 0, 0); //qDebug() << "[] igreenn = " << igreenn;

clone\_oneside\_green.setPixelColor(p, igreenn);

//qDebug() << "[3]";

ibluee = QColor(0, 0, pix\_mask\_blue\_vect.at( (int(sigma) + 1) / 2), 0); //qDebug() << "[] ibluee = " << ibluee;

clone\_oneside\_blue.setPixelColor(p, ibluee);

pix\_mask\_red\_vect.clear();

pix\_mask\_green\_vect.clear();

pix\_mask\_blue\_vect.clear();

}

}

}

ui->label\_7->setPixmap(QPixmap::fromImage(clone\_oneside\_red));

ui->label\_8->setPixmap(QPixmap::fromImage(clone\_oneside\_green));

ui->label\_9->setPixmap(QPixmap::fromImage(clone\_oneside\_blue));

iImage.append(clone\_oneside\_red);

iImage.append(clone\_oneside\_green);

iImage.append(clone\_oneside\_blue);

//Перебираем три созданных изображения для формирования результирующего цветного

for (int i = 0; i < iheight; i++)

{

for(int j = 0; j < iwidth; j++)

{

p = QPoint(j, i);

iredd = clone\_oneside\_red.pixelColor(p);

ired = iredd.red(); //qDebug() << "[] ired = " << ired;

igreenn = clone\_oneside\_green.pixelColor(p);

igreen = igreenn.green(); //qDebug() << "[] ired = " << ired;

ibluee = clone\_oneside\_blue.pixelColor(p);

iblue = ibluee.blue(); //qDebug() << "[] ired = " << ired;

iRgb = QColor(ired, igreen, iblue, 0);

clone\_oneside.setPixelColor(p, iRgb);

}

}

ui->label\_3->setPixmap(QPixmap::fromImage(clone\_oneside));

iImage.append(clone\_oneside);

}

connect(ui->pushButton, SIGNAL(clicked() ), this, SLOT(saveImages() ));

}

ShumForm::~***ShumForm***()

{

delete ui;

}

void ShumForm::**saveImages**()

{

QString saveWay = ui->lineEdit->text(); qDebug() << "[] saveWay = " << saveWay;

QVector <bool> checkboxState;

//Создание массива значение чекбоксов

checkboxState.append(ui->checkBox->isChecked() );

checkboxState.append(ui->checkBox\_2->isChecked() );

checkboxState.append(ui->checkBox\_4->isChecked() );

checkboxState.append(ui->checkBox\_5->isChecked() );

checkboxState.append(ui->checkBox\_6->isChecked() );

checkboxState.append(ui->checkBox\_7->isChecked() );

checkboxState.append(ui->checkBox\_8->isChecked() );

checkboxState.append(ui->checkBox\_9->isChecked() );

checkboxState.append(ui->checkBox\_3->isChecked() );qDebug() << "[] Состояние чекбоксов = " << checkboxState;

bool currentCheckbox = false;

QString str = "AfterProgram-";

QString currentStr, status;

//Перебор массива и сохранение отмеченных картинок

for (int i = 0; i < 9; i++)

{

currentCheckbox = checkboxState.at(i);

if (currentCheckbox == true)

{

currentStr = saveWay + str + QString::number(i) + ".bmp";

iImage.at(i).save( currentStr, "BMP" );

status = status + currentStr + "\n"; qDebug() << status;

}

}

ui->label\_15->setText(status);

qDebug() << "Проверка, кнопка сохранения нажата!";

}

*Листинг mainwindow.ui*

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>247</width>

<height>384</height>

</rect>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<widget class="QWidget" name="centralWidget">

<widget class="QLineEdit" name="lineEdit">

<property name="geometry">

<rect>

<x>10</x>

<y>10</y>

<width>221</width>

<height>20</height>

</rect>

</property>

</widget>

<widget class="QPushButton" name="pushButton\_2">

<property name="geometry">

<rect>

<x>10</x>

<y>60</y>

<width>221</width>

<height>23</height>

</rect>

</property>

<property name="text">

<string>Показать Обрабоку униполярного шума</string>

</property>

</widget>

<widget class="QPushButton" name="pushButton\_3">

<property name="geometry">

<rect>

<x>10</x>

<y>90</y>

<width>221</width>

<height>23</height>

</rect>

</property>

<property name="text">

<string>Показать Обрабоку биполярного шума</string>

</property>

</widget>

<widget class="QLineEdit" name="lineEdit\_2">

<property name="geometry">

<rect>

<x>10</x>

<y>120</y>

<width>51</width>

<height>20</height>

</rect>

</property>

</widget>

<widget class="QLabel" name="label">

<property name="geometry">

<rect>

<x>70</x>

<y>110</y>

<width>161</width>

<height>41</height>

</rect>

</property>

<property name="text">

<string>Коэф. обработки сигма

(формат [-] x.x)</string>

</property>

</widget>

<widget class="QLineEdit" name="lineEdit\_3">

<property name="geometry">

<rect>

<x>10</x>

<y>150</y>

<width>51</width>

<height>20</height>

</rect>

</property>

</widget>

<widget class="QLabel" name="label\_2">

<property name="geometry">

<rect>

<x>70</x>

<y>150</y>

<width>161</width>

<height>41</height>

</rect>

</property>

<property name="text">

<string>Маска

(нечетное целое положитель-

ное число)</string>

</property>

</widget>

<widget class="QLabel" name="label\_3">

<property name="geometry">

<rect>

<x>10</x>

<y>40</y>

<width>221</width>

<height>21</height>

</rect>

</property>

<property name="text">

<string>Фильтр среднего контргармонического</string>

</property>

</widget>

<widget class="QLineEdit" name="lineEdit\_4">

<property name="geometry">

<rect>

<x>10</x>

<y>280</y>

<width>51</width>

<height>20</height>

</rect>

</property>

</widget>

<widget class="QPushButton" name="pushButton\_5">

<property name="geometry">

<rect>

<x>10</x>

<y>250</y>

<width>221</width>

<height>23</height>

</rect>

</property>

<property name="text">

<string>Показать Обрабоку биполярного шума</string>

</property>

</widget>

<widget class="QLabel" name="label\_6">

<property name="geometry">

<rect>

<x>10</x>

<y>200</y>

<width>221</width>

<height>21</height>

</rect>

</property>

<property name="text">

<string>Медианный фильтр:</string>

</property>

</widget>

<widget class="QPushButton" name="pushButton\_4">

<property name="geometry">

<rect>

<x>10</x>

<y>220</y>

<width>221</width>

<height>23</height>

</rect>

</property>

<property name="text">

<string>Показать Обрабоку униполярного шума</string>

</property>

</widget>

<widget class="QLabel" name="label\_5">

<property name="geometry">

<rect>

<x>70</x>

<y>280</y>

<width>161</width>

<height>41</height>

</rect>

</property>

<property name="text">

<string>Маска

(нечетное целое положитель-

ное число)</string>

</property>

</widget>

</widget>

<widget class="QToolBar" name="mainToolBar">

<attribute name="toolBarArea">

<enum>TopToolBarArea</enum>

</attribute>

<attribute name="toolBarBreak">

<bool>false</bool>

</attribute>

</widget>

<widget class="QStatusBar" name="statusBar"/>

<widget class="QMenuBar" name="menuBar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>247</width>

<height>21</height>

</rect>

</property>

</widget>

</widget>

<layoutdefault spacing="6" margin="11"/>

<resources/>

<connections/>

</ui>

*Листинг shumform.ui*

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>ShumForm</class>

<widget class="QWidget" name="ShumForm">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>952</width>

<height>819</height>

</rect>

</property>

<property name="windowTitle">

<string>Form</string>

</property>

<widget class="QLabel" name="label">

<property name="geometry">

<rect>

<x>10</x>

<y>30</y>

<width>231</width>

<height>211</height>

</rect>

</property>

<property name="text">

<string/>

</property>

</widget>

<widget class="QLabel" name="label\_2">

<property name="geometry">

<rect>

<x>260</x>

<y>30</y>

<width>231</width>

<height>211</height>

</rect>

</property>

<property name="text">

<string/>

</property>

</widget>

<widget class="QLabel" name="label\_3">

<property name="geometry">

<rect>

<x>510</x>

<y>30</y>

<width>231</width>

<height>211</height>

</rect>

</property>

<property name="text">

<string/>

</property>

</widget>

<widget class="QLabel" name="label\_6">

<property name="geometry">

<rect>

<x>510</x>

<y>280</y>

<width>231</width>

<height>211</height>

</rect>

</property>

<property name="text">

<string/>

</property>

</widget>

<widget class="QLabel" name="label\_5">

<property name="geometry">

<rect>

<x>260</x>

<y>280</y>

<width>231</width>

<height>211</height>

</rect>

</property>

<property name="text">

<string/>

</property>

</widget>

<widget class="QLabel" name="label\_4">

<property name="geometry">

<rect>

<x>10</x>

<y>280</y>

<width>231</width>

<height>211</height>

</rect>

</property>

<property name="text">

<string/>

</property>

</widget>

<widget class="QLabel" name="label\_7">

<property name="geometry">

<rect>

<x>20</x>

<y>530</y>

<width>231</width>

<height>211</height>

</rect>

</property>

<property name="text">

<string/>

</property>

</widget>

<widget class="QLabel" name="label\_8">

<property name="geometry">

<rect>

<x>270</x>

<y>530</y>

<width>231</width>

<height>211</height>

</rect>

</property>

<property name="text">

<string/>

</property>

</widget>

<widget class="QLabel" name="label\_9">

<property name="geometry">

<rect>

<x>520</x>

<y>530</y>

<width>231</width>

<height>211</height>

</rect>

</property>

<property name="text">

<string/>

</property>

</widget>

<widget class="QLabel" name="label\_10">

<property name="geometry">

<rect>

<x>10</x>

<y>10</y>

<width>47</width>

<height>13</height>

</rect>

</property>

<property name="text">

<string>Исходное:</string>

</property>

</widget>

<widget class="QLabel" name="label\_11">

<property name="geometry">

<rect>

<x>270</x>

<y>10</y>

<width>47</width>

<height>13</height>

</rect>

</property>

<property name="text">

<string>Шум:</string>

</property>

</widget>

<widget class="QLabel" name="label\_12">

<property name="geometry">

<rect>

<x>510</x>

<y>10</y>

<width>121</width>

<height>16</height>

</rect>

</property>

<property name="text">

<string>После удаления шума:</string>

</property>

</widget>

<widget class="QCheckBox" name="checkBox">

<property name="geometry">

<rect>

<x>80</x>

<y>250</y>

<width>81</width>

<height>17</height>

</rect>

</property>

<property name="text">

<string>Сохранить</string>

</property>

</widget>

<widget class="QCheckBox" name="checkBox\_2">

<property name="geometry">

<rect>

<x>330</x>

<y>250</y>

<width>81</width>

<height>17</height>

</rect>

</property>

<property name="text">

<string>Сохранить</string>

</property>

</widget>

<widget class="QCheckBox" name="checkBox\_3">

<property name="geometry">

<rect>

<x>590</x>

<y>250</y>

<width>81</width>

<height>17</height>

</rect>

</property>

<property name="text">

<string>Сохранить</string>

</property>

</widget>

<widget class="QCheckBox" name="checkBox\_4">

<property name="geometry">

<rect>

<x>80</x>

<y>510</y>

<width>81</width>

<height>17</height>

</rect>

</property>

<property name="text">

<string>Сохранить</string>

</property>

</widget>

<widget class="QCheckBox" name="checkBox\_5">

<property name="geometry">

<rect>

<x>330</x>

<y>510</y>

<width>81</width>

<height>17</height>

</rect>

</property>

<property name="text">

<string>Сохранить</string>

</property>

</widget>

<widget class="QCheckBox" name="checkBox\_6">

<property name="geometry">

<rect>

<x>590</x>

<y>510</y>

<width>81</width>

<height>17</height>

</rect>

</property>

<property name="text">

<string>Сохранить</string>

</property>

</widget>

<widget class="QCheckBox" name="checkBox\_7">

<property name="geometry">

<rect>

<x>70</x>

<y>760</y>

<width>81</width>

<height>17</height>

</rect>

</property>

<property name="text">

<string>Сохранить</string>

</property>

</widget>

<widget class="QCheckBox" name="checkBox\_9">

<property name="geometry">

<rect>

<x>590</x>

<y>760</y>

<width>81</width>

<height>17</height>

</rect>

</property>

<property name="text">

<string>Сохранить</string>

</property>

</widget>

<widget class="QCheckBox" name="checkBox\_8">

<property name="geometry">

<rect>

<x>330</x>

<y>760</y>

<width>81</width>

<height>17</height>

</rect>

</property>

<property name="text">

<string>Сохранить</string>

</property>

</widget>

<widget class="QLabel" name="label\_13">

<property name="geometry">

<rect>

<x>750</x>

<y>30</y>

<width>161</width>

<height>41</height>

</rect>

</property>

<property name="text">

<string>Отметьте изображения и пропишите путь к сохранению резуьтатаПропишите путь к сохранению

результатов</string>

</property>

</widget>

<widget class="QLineEdit" name="lineEdit">

<property name="geometry">

<rect>

<x>750</x>

<y>80</y>

<width>171</width>

<height>20</height>

</rect>

</property>

</widget>

<widget class="QPushButton" name="pushButton">

<property name="geometry">

<rect>

<x>750</x>

<y>110</y>

<width>171</width>

<height>23</height>

</rect>

</property>

<property name="text">

<string>Сохранить</string>

</property>

</widget>

<widget class="QLabel" name="label\_14">

<property name="geometry">

<rect>

<x>756</x>

<y>170</y>

<width>161</width>

<height>21</height>

</rect>

</property>

<property name="text">

<string>Сохранённые картинки:</string>

</property>

</widget>

<widget class="QLabel" name="label\_15">

<property name="geometry">

<rect>

<x>760</x>

<y>200</y>

<width>161</width>

<height>591</height>

</rect>

</property>

<property name="text">

<string/>

</property>

</widget>

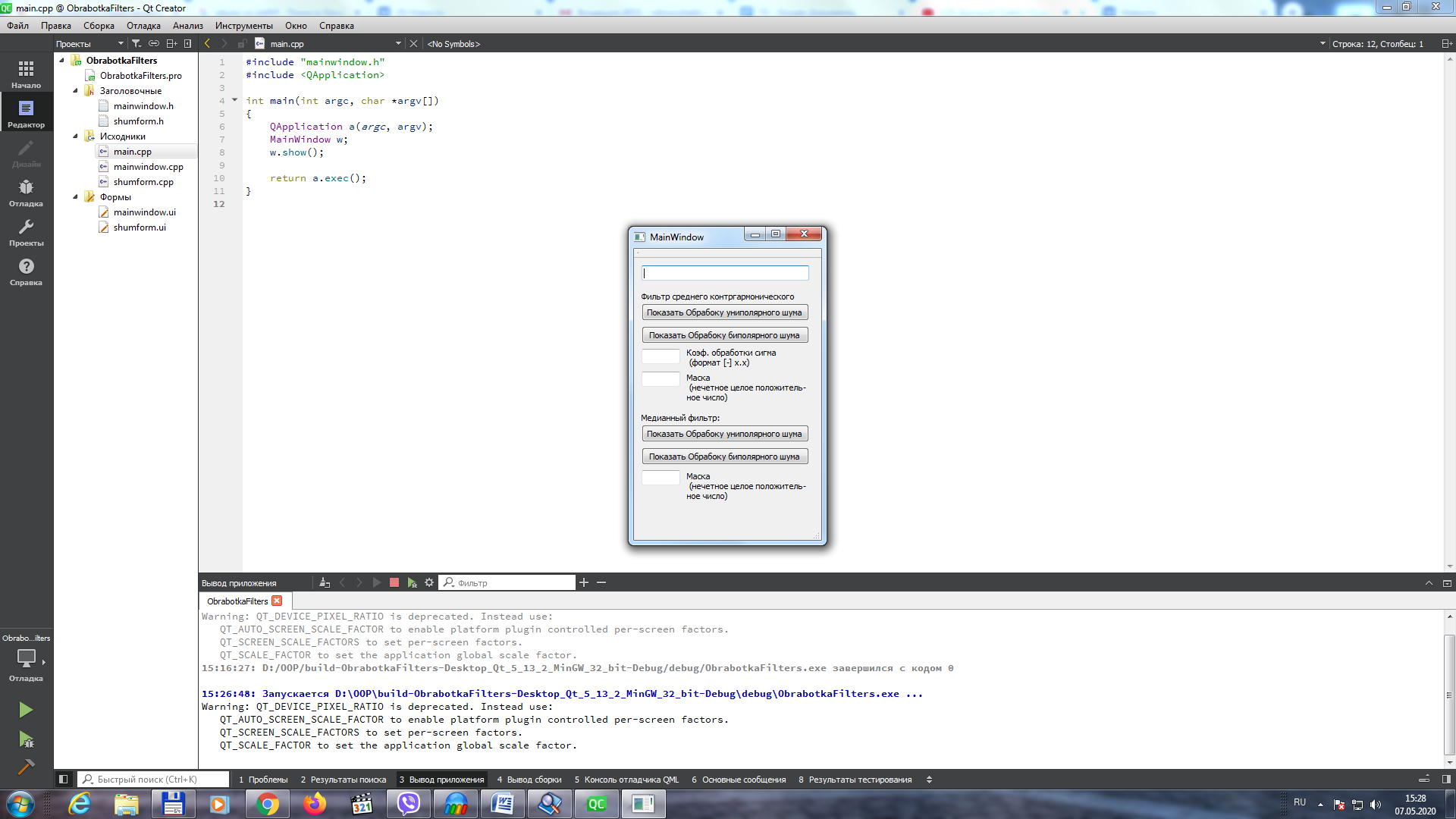
</widget>

<resources/>

<connections/>

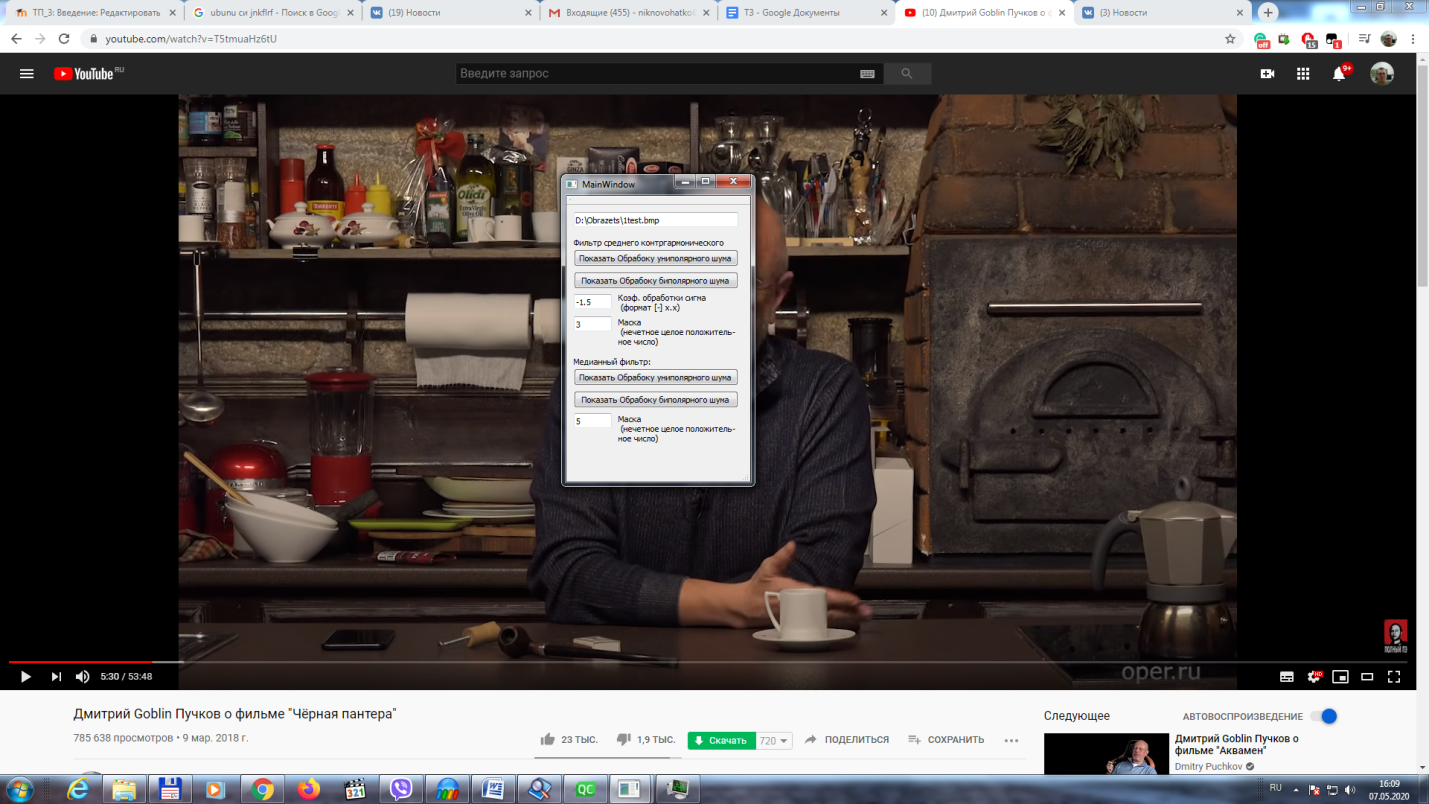
</ui>

*Результат выполнения:*

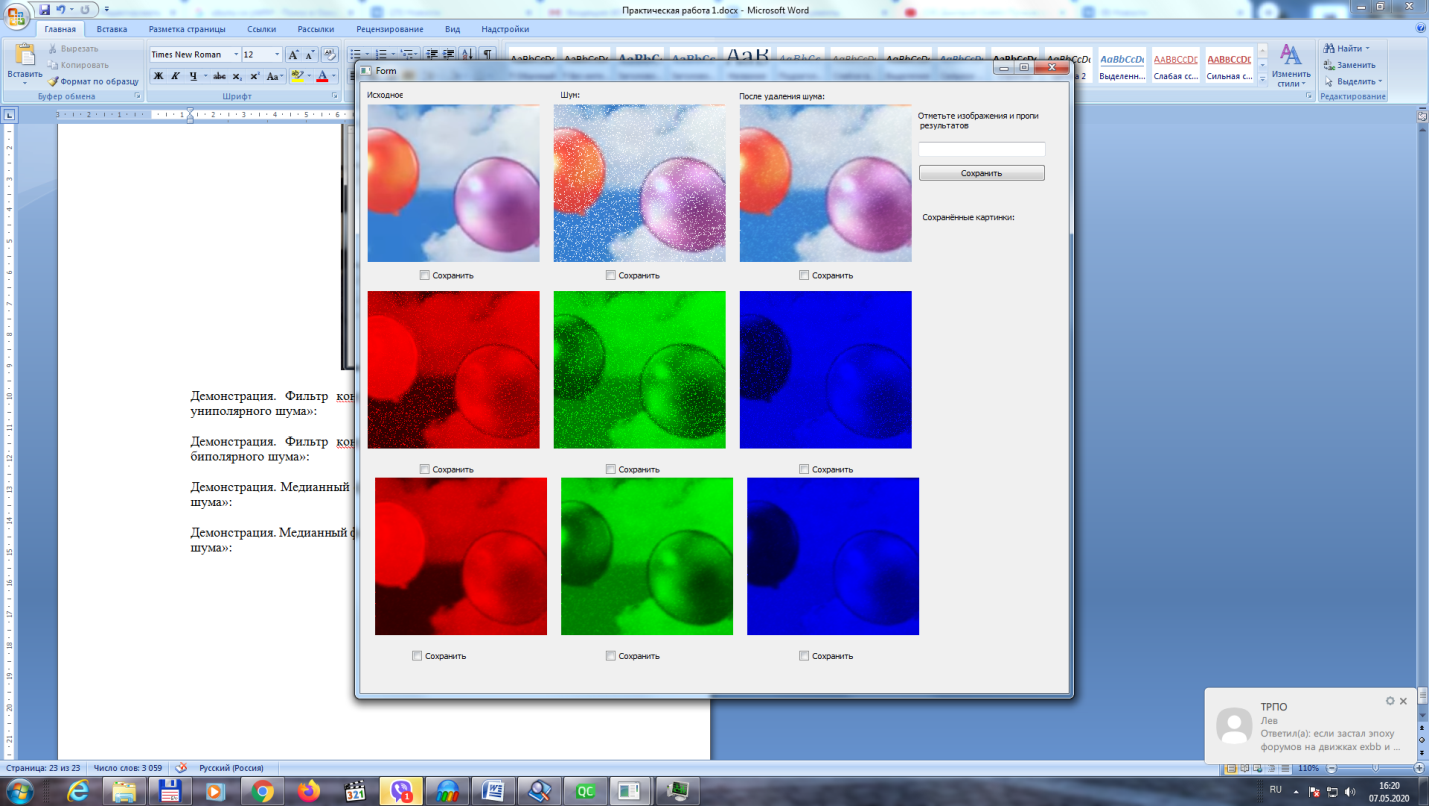
Использование программы:

1. В верхнее поле вносится путь к изображению, на котором будет осуществляться демонстрация. Путь должен быть вида: D:\Obrazets\ 1test.bmp.
2. Параметры для фильтра среднего контргармонического вносятся в поля «Коэф. Обработки» и «Маска» (верхнее поле).
3. Параметр для медианного фильтра вводится в нижнее поле «Маска».
4. Кнопки «Показать обработку униполярного шума» и «Показать обработку биполярного шума» демонстрируют обработку дааных типов шумов для соответствующих фильтров.

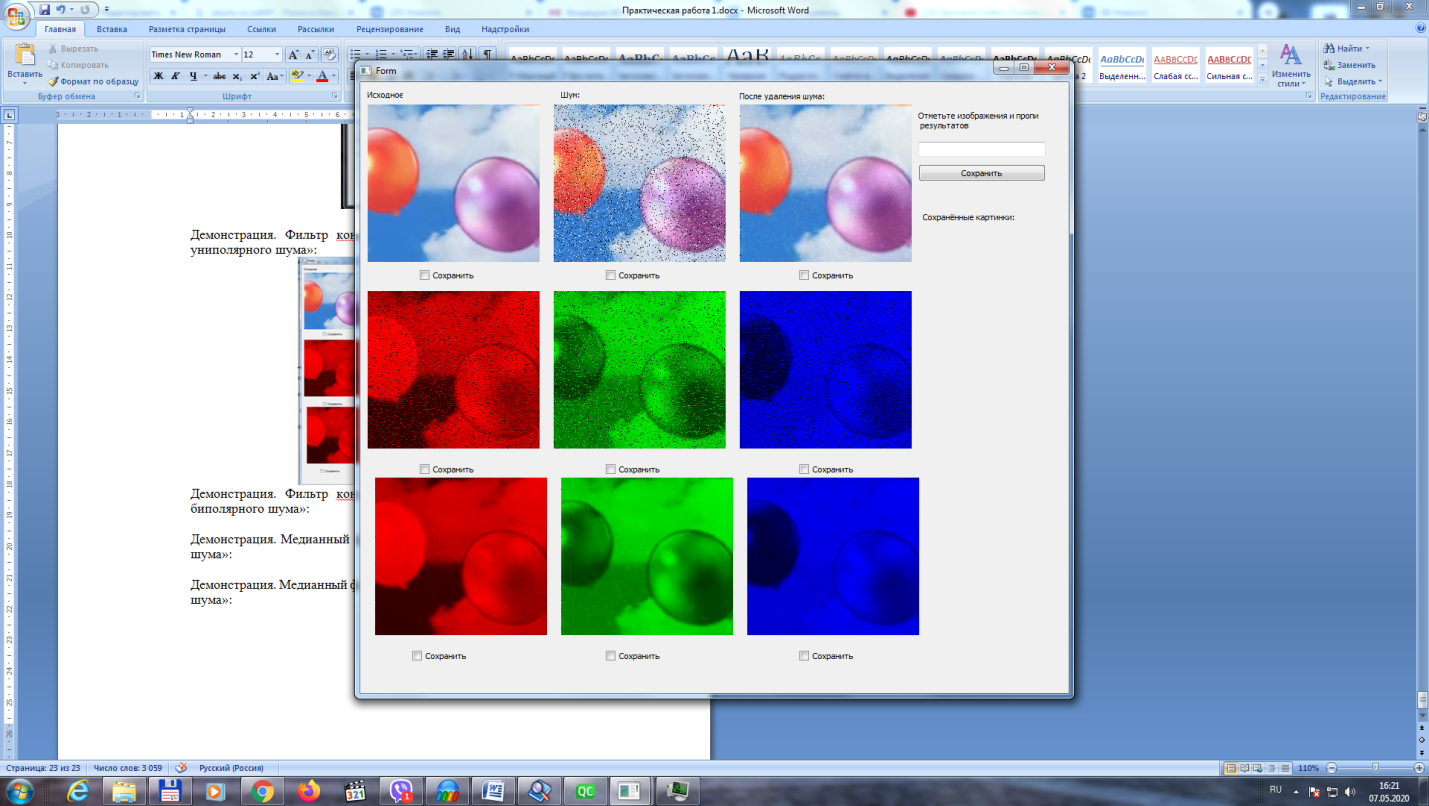
*Демонстрация. Начальные данные:*



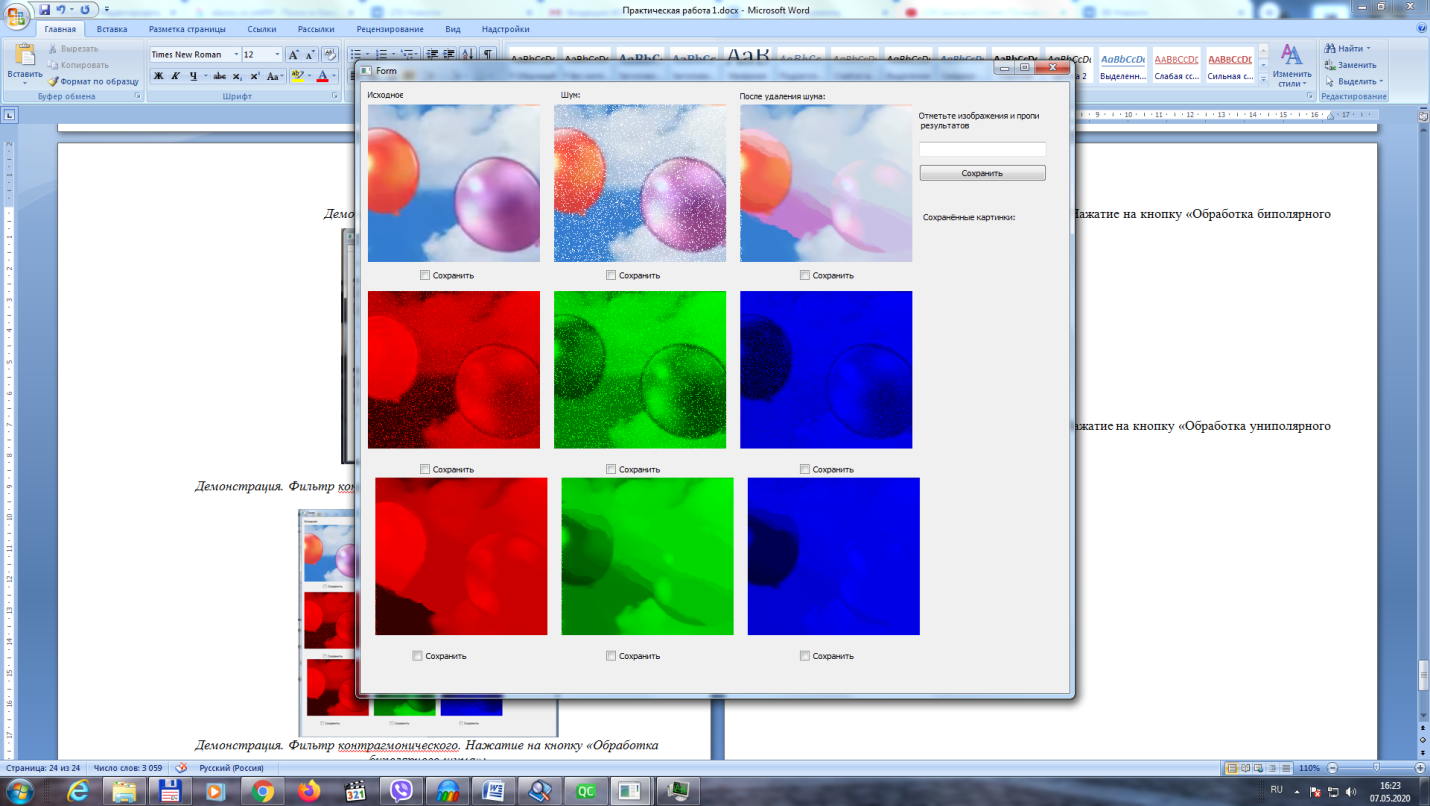
*Демонстрация. Фильтр контрагмонического. Нажатие на кнопку «Обработка униполярного шума»:*



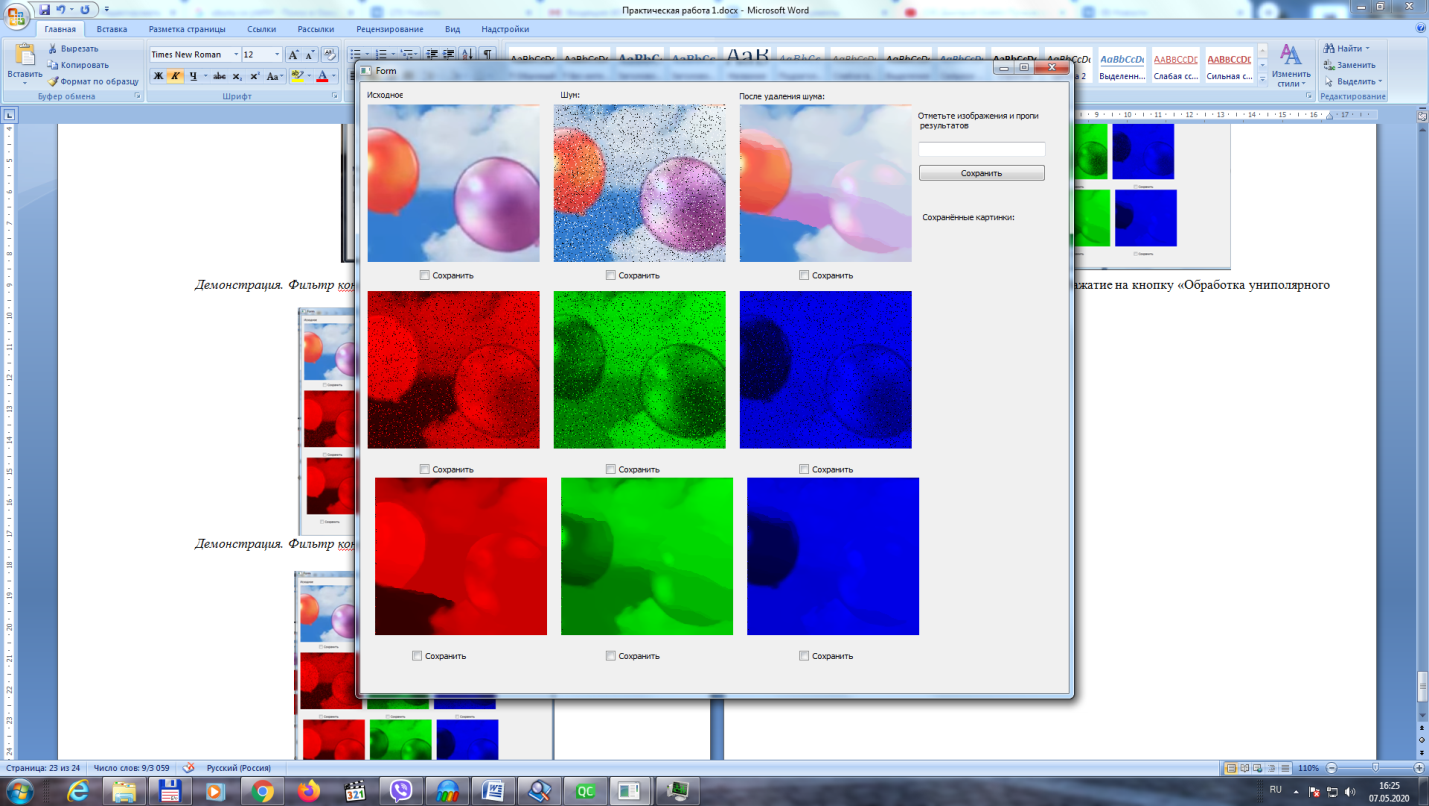
*Демонстрация. Фильтр контрагмонического. Нажатие на кнопку «Обработка биполярного шума»:*



*Демонстрация. Медианный фильтр. Нажатие на кнопку «Обработка биполярного шума»:*



*Демонстрация. Медианный фильтр. Нажатие на кнопку «Обработка униполярного шума»:*



*Сохранение результатов:*

1. Вставляем в поле окна Shum путь к папке, в которую будет осуществляться сохранение.
2. Отмечаем флажками изображения, которые должны быть сохранены.
3. Жмём на кнопку «сохранить», чуть ниже в окне должен появиться список сохраняемых копий.

